2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Model Predictive Trees: Sample-Efficient Receding Horizon Planning with Reusable Tree Search

John Lathrop, Benjamin Rivière, Jedidiah Alindogan, Soon-Jo Chung

***Abstract*— We present Model Predictive Trees (MPT), a receding horizon tree search algorithm that improves its performance by reusing information efficiently. Whereas existing solvers reuse only the highest-quality trajectory from the previous iteration as a "hotstart", our method reuses the entire optimal subtree, enabling the search to be simultaneously guided away from the low-quality areas and towards the high-quality areas. We characterize the restrictions on tree reuse by analyzing the induced tracking error under time-varying dynamics, revealing a tradeoff between the search depth and the timescale of the changing dynamics. In numerical studies, our algorithm outperforms state-of-the-art sampling-based cross-entropy methods with hotstarting. We demonstrate our planner on an autonomous vehicle testbed performing a nonprehensile manipulation task: pushing a target object through an obstacle field. Code associated with this work will be made available at https://github.com/jplathrop/mpt.**

I. Introduction

Gradient-free optimization techniques are an attractive framework for decision making and motion planning on robotic systems where high-fidelity models may not be differentiable and descent algorithms can get caught in local minima. As a motivating example, we consider nonprehensile manipulation, a setting where a robot uses pushing, pulling, or other means of manipulation without grasping the target object with appendages. This task is challenging for conventional gradient-based optimization because of its hybrid dynamics and sparse reward structure.

Upper Confidence Bound for Trees (UCT), a variant of Monte Carlo Tree Search (MCTS), is a powerful gradient-free technique that strategically explores the space of possible future trajectories, with guaranteed convergence to the optimal trajectory as its runtime increases [1]. Although UCT is widely applicable to a large class of decision-making problems, its performance is dependent on its computational resources and accumulating enough samples to make an accurate value estimate. With this context, there is clear benefit in reusing past simulations to improve the quality of search and reduce sample complexity. In this work, we propose an efficient sub-tree recycling procedure and characterize the conditions under changing dynamics where tree recycling is not possible and instead must be recomputed from scratch.

Contributions: We present Model Predictive Trees (MPT), a receding horizon tree-based planner that reuses past subtree information, reducing the cost of searching while greatly boosting the quality of solutions. Saving even low-quality trajectories benefits the search by shifting computational effort from re-generating poor solutions to refining high-quality solutions. We demonstrate our proposed method on a nonprehensile manipulation task performed by an autonomous car in simulation and performing in real-time on onboard hardware. We furthermore provide theoretical guarantees on stability and robustness in the face of changing dynamics. Our method automatically discovers high-level behavior, strategically making and breaking contact with a target object while maneuvering around obstacles to push the target to the goal.

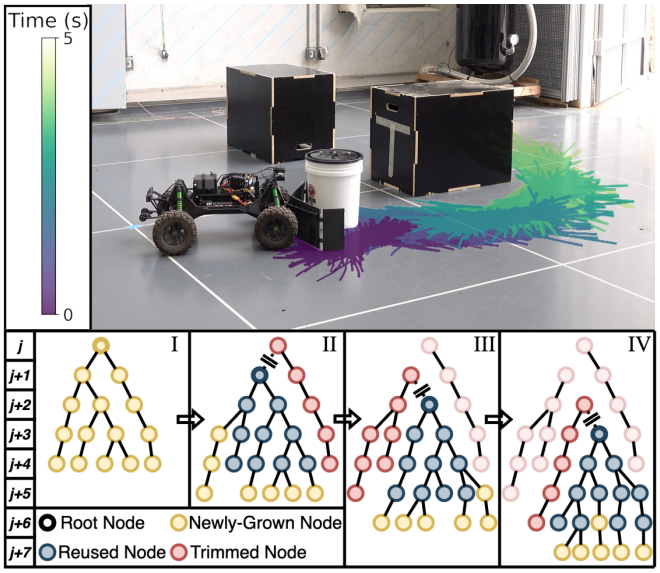

Fig. 1: **Top:** Five seconds of real-time generated trees in our hardware experiment, in which the autonomous vehicle testbed pushes a target to a goal region behind an obstacle. On average, 2100 simulated trajectories are grown every 0.2 s. The trees are colored by the time they were grown. **Bottom:** The proposed tree growth algorithm visualized over four time steps. At each iteration (I - IV), new nodes and branches are added to the tree, and the best first-level child is selected as the next root. In subsequent iterations, older parts of the tree are discarded and new nodes are added.

A. Related Work

Our related work spans sampling-based planning, strategies for reusing data from previous iterations, control-aware planning, and nonprehensile manipulation.

1) Sampling-Based Planners: Sampling-based tree search is common in robotics, with foundational techniques including Rapidly-exploring Random Trees (RRT)) [2], RRT* and variations [3], [4], and Probabilistic Roadmaps (PRM) [5] as well-tested and theoretically-grounded algorithms. These

This work was supported by NSF Grant 2139433 and in part by Supernal.
All authors are with the California Institute of Technology. 1200 E. California Blvd., Pasadena CA 91106, USA {jlathrop,briviere,jedi,sjchung}@caltech.edu

methods use a local planner to connect states and search for a goal region. The stable sparse RRT variation [6] relaxes the need for a local planner.

In contrast, our method uses tree search algorithms investigated in theoretical computer science, in particular the UCT algorithm [1]. UCT and variants have been applied in a variety of problem settings, including robotic task planning [7], motion planning [8], and active sensing [9]. The advantage of using UCT is that our method can plan and execute through hybrid contact dynamics, where the sparse reward and non-smooth contacts make local planning and the restriction to a goal region-objective difficult, therefore limiting the applicability of RRT-based algorithms.

2) Data Reuse in Receding Horizon Planning: Real-time planning is commonly implemented with a receding horizon approach, also known as model-predictive control, where the solver iteratively computes and follows a finite-horizon trajectory until the process is terminated. In this setting, there are various strategies to reuse information from the previous solver iterations in the current iteration. Cross-entropy Motion Planning (CEM) [10] and Model Predictive Path Integral Control (MPPI) [11] are sampling-based methods that initialize the sampling distribution of the next iteration with the mean of the optimal solution from the previous iteration. This approach, sometimes called "hotstarting", averages many simulated trajectories into a summary statistic and throws away a large amount of information.

Deterministic nonlinear model predictive control uses a similar technique to provide an initial guess for the optimization solver [12] with the optimal solution to the previous iteration. Other works have examined tree reuse and tree correction in a control task [13] and with changing obstacles [4].

In contrast to other tree reuse strategies, our method saves the entire selected subtree from the previous iteration, leveraging much more information to refine its search. Our experimental results demonstrate that hotstarting with a subtree provides a larger improvement than hotstarting with the optimal solution, validating our intuition about information reuse. In addition, our reuse approach does not require iteration over the entire tree, a time-consuming requirement that limits real-time deployment.

3) Nonprehensile Manipulation: Although our approach can be applied to a wide class of problems, we focus on nonprehensile manipulation as a motivating example because of its inherent complexity for conventional techniques due to the need to plan through contact. In previous work [14], the authors demonstrate nonprehensile manipulation with a smoothed contact model. In contrast, we do not make a smooth approximation of the contact model, but plan directly in the underlying sparse reward and hybrid dynamical landscape. Whereas other methods use constraints to maintain contact [15], [16], [17], our method does not require contact to be maintained: MPT demonstrates high-level behavior, such as backing-up and re-positioning, where the vehicle maneuvers around the object to execute a better push.

Finally, deep reinforcement learning has been applied to nonprehensile manipulation [18] in simulation. However, this family of methods requires a large amount of offline training data and lacks theoretical guarantees of optimality and stability.

II. Model Predictive Trees

A. Problem Setting

We consider the problem of making decisions over an infinite horizon. For a compact state space $X \subset \mathbb{R}^n$, compact action space $U \subset \mathbb{R}^m$, consider a discrete-time control system characterized by known nominal dynamics $F_{\text{nom}} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and an unknown time-varying disturbance $\mathbf{d} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$, with a bounded reward function $R : \mathbb{R}^n \times \mathbb{R}^m \to [0,1]$. Given an initial condition $\mathbf{x}_0$, the decision-making problem is to maximize the sum of the reward function over an infinite horizon:

$$\mathbf{x}_\infty^*, \mathbf{u}_\infty^* = \underset{\substack{\mathbf{x}_\infty \in \mathbb{R}^\infty \\ \mathbf{u}_\infty \in \mathbb{R}^\infty}}{\arg\max} \sum_{k=1}^{\infty} \gamma^{k-1} R(\mathbf{x}_k, \mathbf{u}_k) \tag{1}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = F_{\text{nom}}(\mathbf{x}_k, \mathbf{u}_{k+1}) + \mathbf{d}(\mathbf{x}_k, \mathbf{u}_{k+1}, k)$$
$$\mathbf{x}_{k+1} \in X, \quad \mathbf{u}_{k+1} \in U, \quad \forall k \in \mathbb{Z}_{\geq 0}$$

where $0 \leq \gamma < 1$ is a discount factor and k is the physical time. The problem (1) is a discounted infinite-horizon Markov Decision Process, given by the tuple $\langle X, U, F, R, D, \gamma \rangle$ with $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ being the nominal dynamics plus disturbance. Manipulation problems are challenging because F is a nondifferentiable function that carries information about the contact between bodies. For example, these dynamics can be modeled with a linear complementarity problem [19]. Here $\mathbf{x}_\infty = [\mathbf{x}_1, \mathbf{x}_2, \ldots]$ denotes an infinite sequence of states, and likewise for $\mathbf{u}_\infty$.

B. Model Predictive Trees (MPT) Method

Our proposed algorithm, specified in Algorithm 1, has two components: a receding horizon UCT-based planner and a contraction-theoretic controller. The planner provides a real-time desired trajectory, and the controller provides exponential stability to the desired trajectory. Moreover, contraction-theoretic control guarantees robust stability and provides a framework to analyze the limitations of tree reuse.

As part of MPT, we incorporate a time-varying estimate of the disturbance $\mathbf{d}$ into the planner. By combining with the user's choice of an online estimator (with some possibilities being adaptive control [20], basis library regression [11], [21], or Bayesian filtering [9]), MPT uses estimates of the disturbance $\hat{\mathbf{d}}_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ to plan over the model:

$$\mathbf{x}_{k+1} = \hat{F}_k(\mathbf{x}_k, \mathbf{u}_{k+1}) = F_{\text{nom}}(\mathbf{x}_k, \mathbf{u}_{k+1}) + \hat{\mathbf{d}}_k(\mathbf{x}_k, \mathbf{u}_{k+1}).$$

The design and stability of the controller have important implications for the accuracy of subtree reuse under changing dynamics, a connection analyzed in Sec. III.

The planning portion of the MPT algorithm (Algorithm 1) performs sequential tree searches in receding horizon fashion, building an incrementally lengthening desired trajectory. We make a distinction between two time indices: physical time is the passing of time in the real-world and is denoted

with k, and simulation time is the time index used in the internal simulation of the tree search and is denoted with j.

At each physical time step $k = 1, 2, \ldots$, MPT performs a search with the UCT algorithm. The search runs a large number of fixed-depth trajectories $\ell = 1, \ldots, L$, approximating the infinite-horizon problem as a K-depth problem with a value estimate $\hat{V}$. The objective of Equation (1) is modified to $\sum_{k=1}^{K} \gamma^{k-1} R(\mathbf{x}_k, \mathbf{u}_k) + \hat{V}(\mathbf{x}_K)$, for $\hat{V} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$.

The collection of rollouts form a decision tree $\mathtt{T}_{k+1}$ that holds information about the cumulative reward (in the manner of Eq. (1)) and number of visits to each node of the tree. The index $k+1$ of $\mathtt{T}_{k+1}$ indicates that the root of the decision tree grown at time k has corresponding time $k+1$.

The exploration-exploitation tradeoff of the UCT algorithm guarantees convergence to the optimal solution of a decision-making problem. Exploration encourages the tree search to investigate new areas of the space of trajectories and exploitation encourages a refinement of the search in parts of the space that have yielded high rewards, with sampled trajectories quickly concentrating to high-valued regions of space. The upper confidence bound formula is as follows, where at each parent node p, the value decides through which child node c to further refine the search:

$$\mathrm{UCT(c)} = \frac{\mathrm{c}.V}{\mathrm{c}.N} + \varepsilon \sqrt{\frac{\log(\mathrm{p}.N)}{\mathrm{c}.N}}, \tag{2}$$

where "$.V$" and "$.N$" refer to the cumulative value and number of visits to a node, respectively. Upon completing L UCT-guided rollouts, the search returns the best action and resulting state out of the search tree $\mathtt{T}_{k+1}$.

The key novelty of our algorithm is the reuse of the selected subtree from the previous iteration to hotstart the tree search at the current time step. When an action $\mathbf{u}_{k+1}$ and corresponding child state $\mathbf{x}_{k+1}$ are selected as "best" out of tree $\mathtt{T}_k$, we trim the search tree $\mathtt{T}_k$ at the connection between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$, keeping the trajectories that start at $\mathbf{x}_{k+1}$. The subtree reuse procedure is shown in Figure 1. Analyzed in Sec. IV, tree reuse enables a more effective search, through which computational power is spent refining high-quality regions of the space of trajectories, rather than re-searching from scratch.

Running in receding horizon fashion, we budget one time step of computational time to the planner, beginning the next iteration's solve with the predicted state $\mathbf{x}_{k+1}$ while the controller is following the trajectory from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$.

To ensure the system arrives near state $\mathbf{x}_{k+1}$ when the tree rooted there is ready, we compose our planner with a contraction-theoretic controller, denoted C in the pseudocode. This controller provides exponential stability to the desired trajectory produced by the planner. We analyze the stability of our proposed control method and the size of steady-state error as a function of the disturbance $\mathbf{d}$ and the disturbance estimates $\hat{\mathbf{d}}$ in Sec. III.

We additionally use a tree reset condition to close the loop on the planning process (Line 14). This condition limits the drift between the simulated tree state and the physical state, resetting the tree at a threshold to ensure the simulated tree state and physical state do not diverge.

As the disturbance may be drifting over time, past subtree information will have used an old estimate $\hat{\mathbf{d}}_{k'}$ that is not up-to-date at the current time k. Re-integrating all trajectories in the tree with the up-to-date dynamics information would require a pass over every single node in the tree, removing the intended benefit of reusing the tree.

Our analysis shows that for slowly-changing dynamics, the steady-state tracking error introduced by the use of past estimates is bounded. Furthermore, understanding the connection between dynamics error, steady-state tracking error, and the contraction metric used to stabilize the system allows us to perform informed hyperparameter tuning. We analyze this connection in Sec. III.

Pseudocode Notes: In Algorithm 1, a node is a tuple ($\mathbf{x}$, $\mathbf{u}$, children, V, N) consisting of a state, the action that led to the state, its list of child nodes, the cumulative value in the subtree below this node, and the total number of visits to this node, respectively. Line 11 is taken to be the real-world application of $\mathbf{u}_{k+1}$. $DARE(A,B,Q,R)$ is the Discrete Algebraic Riccati Equation (7). In Line 12, we put a placeholder for the user's choice of dynamics estimator, which may be a function of time k, the state/input $(\mathbf{x}_k, \mathbf{u}_k)$, desired state/input $(\mathbf{x}_k^d, \mathbf{u}_k^d)$, or other latent variables.

III. Theoretical Results

In this section, we review discrete contraction analysis, derive the robust stability of our controller, and relate the tree reuse to the steady-state error of the proposed controller. Whereas previous work has used contraction theory in planning to stabilize local trajectories to an existing global plan, both in the optimization [22] and learning [23] context, we use contraction theory to analyze the tree reset procedure.

Notations: All norms are the standard 2-norm unless otherwise specified. $\mathtt{I}_p$ denotes the $p \times p$ identity matrix. A square symmetric matrix A is positive definite ($A \succ 0$), positive semidefinite ($A \succeq 0$), negative definite ($A \prec 0$), or negative semidefinite ($A \preceq 0$) if its eigenvalues are positive, nonnegative, negative, or nonpositive, respectively. $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the largest and smallest eigenvalues of A.

A. Discrete-Time Contraction Theory

Consider a discrete-time, time-varying dynamical system

$$\mathbf{q}_{k+1} = F(\mathbf{q}_k, k), \tag{3}$$

for time $k \in \mathbb{Z}_{\geq 0}$, state $\mathbf{q} : \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$, and a bounded transition function $F : \mathbb{R}^n \times \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$.

We consider the infinitesimal variation of our system: $\delta\mathbf{q}_{k+1} = \frac{\partial F}{\partial \mathbf{q}}(\mathbf{q}_k, k)\delta\mathbf{q}_k$. We say the system (3) is *contracting* if all solutions exponentially converge to a single trajectory.

Theorem 1. *A necessary and sufficient condition for (3) to be contracting [24] is the existence of a uniformly positive definite matrix $M(\mathbf{q},k) = \Theta(\mathbf{q},k)^\top \Theta(\mathbf{q},k) \in \mathbb{R}^{n \times n}$, called a*

Algorithm 1: Model Predictive Trees

Parameters: τ : Reset threshold

def Model_Predictive_Trees($\mathbf{x}_0; \langle X, U, F_{\text{nom}}, R, D, \gamma \rangle, C$):
- // create an initial tree
- $\mathtt{T}'_1$.root $= (\mathbf{x}_0, \mathbf{0}, [\,], 0, 0)$;
- $\hat{F}_1 = F_{\text{nom}}$;
- **for** $k = 1, \ldots, \infty$ **do**
  - // build tree
  - $\mathtt{T}_k = \text{UCT\_search}(\mathtt{T}'_k; \langle X, U, \hat{F}_k, R, D, \gamma \rangle)$;
  - // extract desired trajectory
  - $\mathtt{T}_k$.root.best_child $= \arg\max\{\text{c}.V/\text{c}.N\}$
  - for c in $\mathtt{T}_k$.root.children ;
  - $\mathbf{x}^d_k = \mathtt{T}_k$.root.$\mathbf{x}$;
  - $\mathbf{u}^d_k = \mathtt{T}_k$.root.best_child.$\mathbf{u}$;
  - // follow desired trajectory
  - $\mathbf{u}_{k+1} = C(\mathbf{x}_k, \mathbf{x}^d_k, \mathbf{u}^d_{k+1}; \hat{F}_k)$;
  - $\mathbf{x}_{k+1} = \text{rollout}(\mathbf{x}_k, \mathbf{u}_{k+1})$;
  - // estimate disturbance
  - $\hat{\mathbf{d}}_{k+1} = \text{dynamics\_estimate}(\ldots)$;
  - $\hat{F}_{k+1}(\cdot,\cdot) = F_{\text{nom}}(\cdot,\cdot) + \hat{\mathbf{d}}_{k+1}(\cdot,\cdot)$;
  - // trim tree
  - $\mathtt{T}'_{k+1}$.root $= \mathtt{T}_k$.root.best_child ;
  - **if** $\|\mathtt{T}'_{k+1}.\text{root}.\mathbf{x} - \mathbf{x}_{k+1}\| > \tau$ **then**
    - $\mathtt{T}'_{k+1}$.root $= (\mathbf{x}_{k+1}, \mathbf{0}, [\,], 0, 0)$

Search parameters:
L : Number of iterations, b : Branching factor,
K : Search depth, ε : Exploration constant

def UCT_search($\mathtt{T}_k; \langle X, U, F, R, D, \gamma \rangle$):
- **for** $\ell = 1, \ldots, L$ **do**
  - path $= [\mathtt{T}_k.\text{root}]$;
  - // rollout
  - **for** $j = k, \ldots, K + k$ **do**
    - **if** *len*(path$[-1]$.children) $< b$ **then**
      - $\mathbf{u}_{j+1} \sim U$; // sample action
      - $\mathbf{x}_{j+1} = F(\text{path}[-1].\mathbf{x}, \mathbf{u}_{j+1})$; // step
      - next_node $= (\mathbf{x}_{j+1}, \mathbf{u}_{j+1}, [\,], 0, 0)$;
      - path$[-1]$.children.append(next_node)
    - **else**
      - next_node $= \underset{\text{c} \in \text{path}[-1]}{\arg\max} \left\{ \frac{\text{c}.V}{\text{c}.N} + \varepsilon \sqrt{\frac{\log(\text{path}[-1].N)}{\text{c}.N}} \right\}$;
    - path.append(next_node)
  - // backpropagate
  - cumulative_reward $= 0$;
  - **for** node *in* path.reversed() **do**
    - node.N += 1 ;
    - node.V += cumulative_reward ;
    - cumulative_reward $=$ $\gamma \cdot$ cumulative_reward $+ R(\text{node}.\mathbf{x}, \text{node}.\mathbf{u})$
- **return** $\mathtt{T}_k$

Controller parameters:
Q : State cost, R : Input cost

def $C(\mathbf{x}_k, \mathbf{x}^d_k, \mathbf{u}^d_{k+1}; F)$:
- $A = \frac{\partial F}{\partial x}|_{\mathbf{x}^d_k, \mathbf{u}^d_{k+1}}$, $B = \frac{\partial F}{\partial u}|_{\mathbf{x}^d_k, \mathbf{u}^d_{k+1}}$;
- $M = DARE(A, B, Q, R)$;
- $K = (B^\top M B + R)^{-1} B^\top M A$;
- $\mathbf{u} = \mathbf{u}^d_{k+1} - K(\mathbf{x}_k - \mathbf{x}^d_k)$;
- **return** $\mathbf{u}$

contraction metric, where Θ defines a smooth and invertible coordinate transformation $\delta\mathbf{p} = \Theta(\mathbf{q}, k)\delta\mathbf{q}$, which $\forall \mathbf{q}, k$:

$$\frac{\partial F}{\partial \mathbf{q}}(\mathbf{q}, k)^\top M(\mathbf{q}, k) \frac{\partial F}{\partial \mathbf{q}}(\mathbf{q}, k) - \alpha^2 M(\mathbf{q}, k) \preceq 0, \tag{4}$$

for constant $0 \leq \alpha < 1$, called the contraction rate.

Contraction analysis also extends to discrete-time systems with disturbances. Consider now a perturbed system:

$$\mathbf{q}_{k+1} = F(\mathbf{q}_k, k) + \sigma(k). \tag{5}$$

Theorem 2. *Let the system (3) be a contracting system with metric $M = \Theta^\top \Theta$, contraction rate $0 \leq \alpha < 1$, and a particular solution $\mathbf{x}_k$. If $\mathcal{L}^\infty(\Theta\sigma) < \infty$, then any solution $\mathbf{z}_k$ to the perturbed system (5) converges exponentially to an error ball around $\mathbf{x}_k$. Furthermore, if we assume M is uniformly bounded as $\underline{m}\mathtt{I}_n \preceq M \preceq \overline{m}\mathtt{I}_n$ and $\mathcal{L}^\infty(\sigma) \leq \overline{\sigma}$, the solutions satisfy, for all k:*

$$\|\mathbf{z}_k - \mathbf{x}_k\| \leq \alpha^k \sqrt{\frac{\overline{m}}{\underline{m}}} \|\mathbf{z}_0 - \mathbf{x}_0\| + \frac{\overline{\sigma}}{1-\alpha} \sqrt{\frac{\overline{m}}{\underline{m}}} (1 - \alpha^k).$$

Proof. The proof is shown in [24]. □

B. Exponential Convergence to a Desired Trajectory

With the tools of discrete-time contraction, we proceed to analyze the tracking performance of our proposed algorithm. So far in the analysis we have considered dynamical systems without control inputs; we now present a constructive proof for a feedback law that guarantees the contraction of a discrete-time control system:

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_{k+1}). \tag{6}$$

The proposed controller is a locally-linearized Ricatti controller that, under suitable assumptions, stabilizes the system to a desired trajectory $(\mathbf{x}^d_k, \mathbf{u}^d_k)$ for $k \in \mathbb{Z}_{\geq 0}$.

Assumption 1. *For positive definite cost matrices Q and R, we assume the linearized system at each k given by*

$$A(\mathbf{x}^d_k, \mathbf{u}^d_{k+1}) = \nabla_{\mathbf{x}} F|_{(\mathbf{x}^d_k, \mathbf{u}^d_{k+1})}, \; B(\mathbf{x}^d_k, \mathbf{u}^d_{k+1}) = \nabla_{\mathbf{u}} F|_{(\mathbf{x}^d_k, \mathbf{u}^d_{k+1})},$$

satisfies (A, B) are stabilizable and $(A, Q^{\frac{1}{2}})$ are observable. This guarantees a unique positive definite solution exists to the Discrete Algebraic Riccati Equation (DARE) [25]:

$$M = A^\top M A - (A^\top M B)(R + B^\top M B)^{-1}(B^\top M A) + Q. \tag{7}$$

Furthemore, we assume the solution to DARE (7) is uniformly bounded over the state space as $\underline{m}\mathtt{I}_n \preceq M \preceq \overline{m}\mathtt{I}_n$.

Theorem 3. *Under Assumption 1, consider the feedback law*

$$\mathbf{u}_{k+1} = \mathbf{u}^d_{k+1} - K(\mathbf{x}_k - \mathbf{x}^d_k), \tag{8}$$

with $K = (R + B^\top M B)^{-1}(B^\top M A)$ and M the solution to $DARE(A, B, Q, R)$ (7). The feedback law in (8) yields a closed-loop system that contracts to $\mathbf{x}^d_k$ with metric M and rate $1 > \alpha \geq \sqrt{1 - \frac{\lambda_{min}(Q)}{\overline{m}}}$.

Proof. Our goal is to show that, for $A_{cl} = A - BK$, $A_{cl}^\top M A_{cl} - \alpha^2 M \preceq 0$ holds $\forall \mathbf{x}, k$. Manipulating,

$$A_{cl}^\top M A_{cl} - \alpha^2 M = (A-BK)^\top M (A-BK) - \alpha^2 M$$
$$= A^\top MA - A^\top MBK - K^\top B^\top MA + K^\top B^\top MBK - \alpha^2 M.$$

Plugging in the definition of K, note that $K^\top B^\top MBK = K^\top B^\top MA - K^\top RK$, and consequently, the above becomes

$$= A^\top MA - A^\top MBK - K^\top RK - \alpha^2 M$$
$$= A^\top MA - A^\top MB(R+B^\top MB)^{-1}(B^\top MA) - K^\top RK - \alpha^2 M.$$

As M solves $DARE(A,B,Q,R)$, the above becomes

$$= M - Q - K^\top RK - \alpha^2 M = (1-\alpha^2)M - Q - K^\top RK,$$

where $(1-\alpha^2)M - Q - K^\top RK \preceq 0$ holds if $(1-\alpha^2)M - Q \preceq 0$, which holds if $(1-\alpha^2)\overline{m} - \lambda_{\min}(Q) \leq 0$. Thus, the system is contracting with rate $\alpha \geq \sqrt{1 - \frac{\lambda_{\min}(Q)}{\overline{m}}}$. □

Remark 1. *We note that the linearizations in the controller are made about the desired trajectory $(\mathbf{x}_k^d, \mathbf{u}_{k+1}^d)$, but if Assumption 1 holds for the true state and desired input $(\mathbf{x}_k, \mathbf{u}_{k+1}^d)$, we can linearize there.*

The proposed feedback law also enjoys a straightforward robustness result. Consider a trajectory $(\mathbf{x}_k^d, \mathbf{u}_k^d)$ that is a solution to the system (6), and consider the disturbed system:

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_{k+1}) + \sigma(k). \tag{9}$$

Lemma 1. *Under Assumption 1, the proposed control law renders the closed-loop system (9) exponentially stable to an error ball around the desired trajectory:*

$$\|\mathbf{x}_k - \mathbf{x}_k^d\| \leq \alpha^k \sqrt{\frac{\overline{m}}{\underline{m}}} \|\mathbf{x}_0 - \mathbf{x}_0^d\| + \frac{\overline{\sigma}}{1-\alpha} \sqrt{\frac{\overline{m}}{\underline{m}}} (1-\alpha^k),$$

where $\mathcal{L}^\infty(\sigma) \leq \overline{\sigma}$ and $\alpha = \sqrt{1 - \frac{\lambda_{min}(Q)}{\overline{m}}}$.

Proof. The proof follows from the application of Theorem 2 to the closed-loop system in Theorem 3. □

We apply this robustness result by analyzing the error introduced by reusing subtrees as the disturbance changes over time. Past subtrees will have been built with an old estimate, introducing error in the difference between the past and current dynamics estimates. We first characterize the effect of a time-varying disturbance.

Assumption 2. *Consider the disturbance $\mathbf{d}$ in Equation (1). We suppose that $\mathbf{d}$ is "slowly changing": that the temporal difference of $\mathbf{d}$ is bounded. Furthermore, we assume the dynamics estimate available to the algorithm at a time k is ε-accurate. There exists $\eta, \varepsilon \in \mathbb{R}_+$ such that for all $\mathbf{x}, \mathbf{u}, k$:*

$$\|\mathbf{d}(\mathbf{x},\mathbf{u},k+1) - \mathbf{d}(\mathbf{x},\mathbf{u},k)\| \leq \eta$$
$$\|\mathbf{d}(\mathbf{x},\mathbf{u},k) - \hat{\mathbf{d}}_k(\mathbf{x},\mathbf{u})\| \leq \varepsilon,$$

Under these assumptions, we can quantify the error introduced by reusing incorrect information from the past and understand the effect it has on the tracking error.

Theorem 4. *Under Assumptions 1-2, the steady-state tracking error in MPT is bounded as:*

$$\|\mathbf{x}_\infty - \mathbf{x}_\infty^d\| \leq \sqrt{\frac{\overline{m}}{\underline{m}}} \frac{(K+1)\eta + \varepsilon}{1-\alpha}, \tag{10}$$

for K the depth of the tree search and $\alpha = \sqrt{1 - \frac{\lambda_{min}(Q)}{\overline{m}}}$.

Proof. At time step i, with corresponding dynamics estimate $\hat{\mathbf{d}}_i$, the trajectories in the search run for simulation time $j = i+1, \ldots, i+K+1$. As such, the maximal time difference between a dynamics estimate and when a tree branch using that dynamics estimate becomes part of the desired trajectory is $K+1$ time steps. Therefore, when the physical time is $k = i+K+1$, by Assumption 2,

$$\|\mathbf{d}(\mathbf{x},\mathbf{u},i+K+1) - \hat{\mathbf{d}}_i(\mathbf{x},\mathbf{u})\| \leq (K+1)\eta + \varepsilon.$$

As the desired trajectory at $k = i+K+1$ satisfies the dynamics $\mathbf{x}_{k+1} = F_{\text{nom}}(\mathbf{x}_k, \mathbf{u}_{k+1}) + \hat{\mathbf{d}}_i(\mathbf{x}_k, \mathbf{u}_{k+1})$ and the actual rollout satisfies $\mathbf{x}_{k+1} = F_{\text{nom}}(\mathbf{x}_k, \mathbf{u}_{k+1}) + \mathbf{d}(\mathbf{x}_k, \mathbf{u}_{k+1}, k)$, the tracking error of following the desired trajectory is:

$$\|\mathbf{x}_k - \mathbf{x}_k^d\| \leq \alpha^k \sqrt{\frac{\overline{m}}{\underline{m}}} \|\mathbf{x}_0 - \mathbf{x}_0^d\| + \frac{(K+1)\eta + \varepsilon}{1-\alpha} \sqrt{\frac{\overline{m}}{\underline{m}}} (1-\alpha^k).$$

Letting $k \to \infty$ yields a steady-state tracking error:

$$\|\mathbf{x}_\infty - \mathbf{x}_\infty^d\| \leq \sqrt{\frac{\overline{m}}{\underline{m}}} \frac{(K+1)\eta + \varepsilon}{1-\alpha}.$$ □

The expression (10) dictates the limit of tree reuse in the presence of changing dynamics. With this understanding of how the depth of the tree search affects the steady-state tracking error, we can conduct informed parameter design when planning our tree search. For a given steady-state error threshold, a tradeoff exists between how quickly the disturbance is changing (given by η) and how far in the future we can search with tree reuse.

IV. Experimental Results and Discussion

We demonstrate MPT on an autonomous vehicle testbed in simulation and hardware.

A. Experimental Setup

In our experiments, our algorithm solves a planar nonprehensile manipulation task, with reward given for pushing a cylindrical object (a barrel) to a goal position. Let the state be $\mathbf{x} = [x, y, \theta, x_o, y_o]^\top \in \mathbb{R}^5$, where (x,y) is the inertial position of the vehicle in meters, θ is the heading in radians, and (x_o, y_o) is the position of the center of the barrel in meters. The control inputs are $\mathbf{u} = [V, \delta]^\top \in \mathbb{R}^2$, where V is speed in meters per second and δ is steering angle in radians, describing an Ackermann car. The nonlinear dynamics are:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta t \begin{bmatrix} V_{k+1}\cos(\theta_k) \\ V_{k+1}\sin(\theta_k) \\ \frac{V_{k+1}}{l}\tan(\delta_{k+1}) \end{bmatrix}$$

$$\begin{bmatrix} x_{o,k+1} \\ y_{o,k+1} \end{bmatrix} = \text{LCP}\left(\begin{bmatrix} x_{o,k} \\ y_{o,k} \end{bmatrix}, \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} \right)$$

with Δt being the time step and l the wheelbase length. The states x_o, y_o are found by numerically solving the non-penetration constraints with respect to the car geometry in Fig. 2 as a linear complementarity problem (LCP), as in [19].

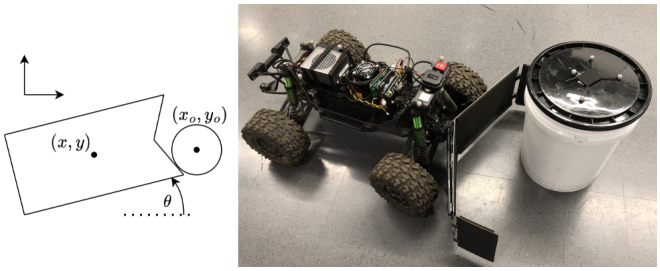

Fig. 2: **Left:** The states and collision geometry of the simulation model used in the experiments. **Right:** The autonomous vehicle platform and barrel equipped with sensors for state estimation and compute for running our algorithm.

The reward function is the sum of a nominal reward and a term that increases as the object position approaches (x_g, y_g):

$$R(\mathbf{x}, \mathbf{u}) = 0.1 + 0.9\left(1 - \frac{1}{D}\sqrt{(x_o - x_g)^2 + (y_o - y_g)^2}\right)$$

for normalizing constant D. This reward is sparse because when the vehicle is not in contact with the barrel, no improvement in reward is available until first, contact is made and second, the barrel is pushed toward the goal.

The input limits are $|V| \leq 1$ m/s, $|\delta| \leq 0.42$ rad, according to the steering limits of our platform. For MPT (and the UCT baseline below), we discretize the action space as $(V, \delta) \in \{(0,0), (\pm 1, 0), (\pm 1, \pm 0.42)\}$, sampling uniformly without replacement during tree growth.

B. Baselines

We compare our method against 3 baselines: (1) UCT deployed with no tree reuse, referred to as "UCT" in our experiments. This algorithm operates in the same way as MPT, but the next root node contains no children from the previous iteration; (2) a cross-entropy motion planner (CEM) implemented based on [10] with a Gaussian input distribution, ten iterations, and 10% elite particle fraction; and (3) CEM that hotstarts sampling with the optimal solution of the previous iteration (CEM-Reuse).

C. Numerical Experiments

In Fig. 3, we compare the performance of MPT against the baselines of UCT, CEM, and CEM-Reuse on a grid of initial states. For $\theta_0 = 0$, $x_{o,0} = y_{o,0} = 0$, we vary the initial x and y position of the car over a 4m $\times$ 4m space. The goal is to push the barrel from its initial position at $(0,0)$ to $(x_g, y_g) = (4,0)$. Here, the value is calculated as the realized (undiscounted) cumulative reward of running each planner in receding horizon fashion for 100 time steps, executing the first proposed action and replanning at each time step.

The value produced by each method averaged over the state space is summarized below. Information reuse results in a significant improvement between UCT and MPT. Whereas

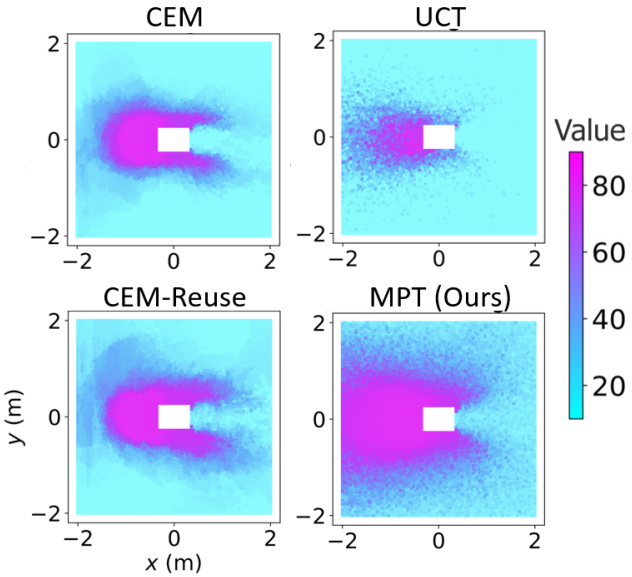

Fig. 3: For a grid of (x, y) initial car positions, we color each point according to the accumulated value of running each algorithm (CEM, CEM-Reuse, UCT, MPT). Purple indicates higher value. These simulations were generated with $L = 200$, a planning horizon of 10, and a simulation depth of 100. The value shown is averaged across ten runs at each initial condition. Our proposed method, MPT, provides the best average cumulative reward of all methods, with a significant improvement over the baseline UCT method.

UCT is outperformed by CEM, the improvement due to reusing information results in MPT having a 29.9% higher value than CEM-Reuse, the next best method.

Method	**CEM**	**CEM-Reuse**	**UCT**	**MPT (Ours)**
Average Value	23.63	28.97	16.79	37.64
Reuse % Improvement	-	22.6%	-	124%

In the task, CEM methods are unable to reliably find a solution unless the car is initialized close to the barrel. Each method performs most consistently when the vehicle starts directly to the left of the barrel, where driving forward will push the barrel to the goal.

MPT is able to find high-valued solutions even when the initial position is far from the barrel. MPT can quickly find these "needle in a haystack" solutions that require a coordinated maneuver to make contact with then push the barrel to the goal. The reuse of the search trees of previous iterations allows MPT to quickly concentrate its search on high-valued trajectories, without wasting computational effort re-searching through low-valued trajectories.

D. Sample Efficiency

We examine sample efficiency by considering one initial condition $\begin{bmatrix} x & y & \theta & x_o & y_o \end{bmatrix}^\top = \begin{bmatrix} -1.5 & -0.5 & 0 & 0 & 0 \end{bmatrix}^\top$ and goal position $(x_g, y_g) = (4,0)$. We deploy each algorithm in receding horizon fashion where at each time step, L simulations are run and the first step of the plan is taken. As before, we

evaluate the cumulative reward. We visualize this metric in Fig. 4 vs. the number of simulations L.

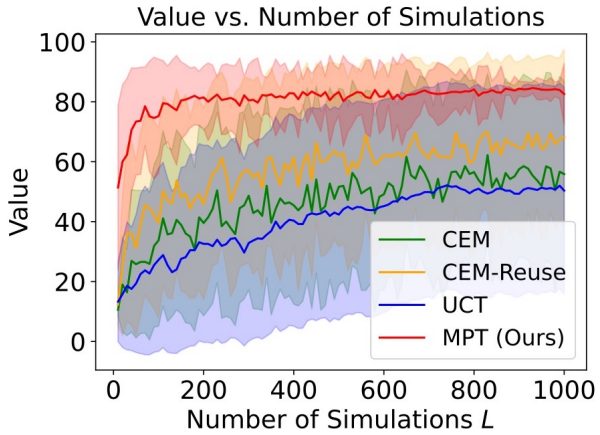

Fig. 4: The value of the trajectory produced by each planning method versus the number of simulations. For each L, 100 trials are run, and the average value is plotted with one standard deviation error bar. Our proposed algorithm (MPT) significantly outperforms the baselines and has a less noisy estimate.

Our proposed algorithm greatly outperforms the baselines, with a rapid rise in value, reaching an asymptotic limit at $L = 180$. The competing baselines exhibit a much slower increase in value, highlighting the sample efficiency of MPT. Furthermore, the value estimates produced by the baselines are significantly noisier than that of MPT. We extend the simulation count to see where the average value produced by each baseline draws level to the asymptotic limit found by MPT, with UCT not catching up in the considered range.

Method	MPT	CEM	CEM-Reuse	UCT
L needed to reach $V = 80$	180	5200	3000	>30000

E. Hardware Results

We verify the ability of MPT to be deployed on hardware by implementing our algorithm on the autonomous vehicle testbed shown in Fig. 2. We task MPT to solve the barrel-pushing task in an environment with three obstacles. MPT is able to plan in real time and execute a 12−second pushing operation that maneuvers the barrel around the obstacles to the goal position. The trajectory of the vehicle and the barrel around the obstacles are shown in Fig. 5.

Our algorithm is able to plan through high-level behaviors of making and breaking contact with the barrel. Halfway through the experiment, the vehicle stops, backs up, and re-positions itself behind the barrel to make a push around an obstacle. Such a maneuver is only possible if planning through the hybrid dynamics, a distinct advantage of our proposed method. We show that state-of-the-art baselines are either too sample-inefficient or unable to plan through the dynamics, rendering the observed behavior unique to MPT.

At each planning iteration, we check the tree reset condition with $\tau = 0.5$. At three times in the experiment, the tree is reset when the state of the vehicle or barrel diverge

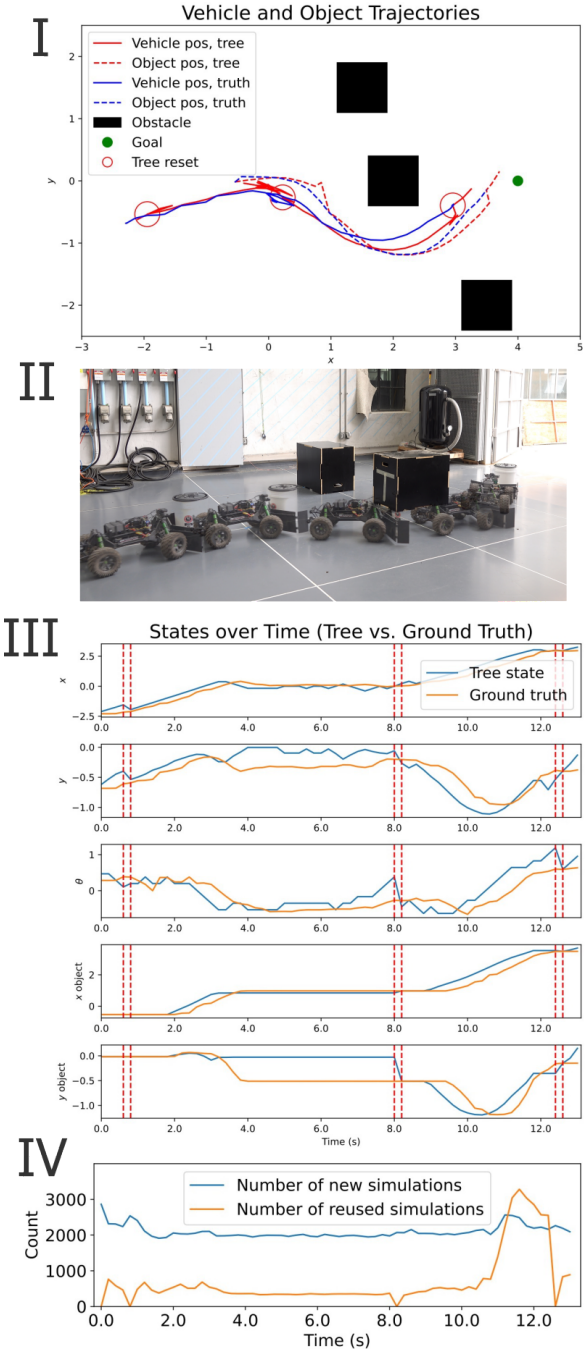

Fig. 5: Our algorithm plans for and executes a solution onboard an autonomous vehicle testbed. **I:** The trajectory of the vehicle (solid line) and the pushed object (dashed line) are shown, with the planned trajectory in red and the actual trajectory in blue. The tree resets are circled in red. **II:** An overlay of the trajectory of the vehicle and barrel over the course of the experiment in the Caltech Center for Autonomous Systems and Technologies. **III:** The states as simulated by the tree (blue) and as measured by the motion capture (orange). The tree reset instances are each shown as a pair of red lines. **IV:** The number of simulations saved by reusing the tree is shown. On average, one third of the new simulations are carried over to the next planning iteration. Near the end of the experiment, when the optimal behavior is easy to find, the number of reused simulations rises dramatically. The decision trees grown here are highly concentrated to the optimal actions at each depth.

from the simulated state. The resets are all triggered by a mismatch in the θ state or the object position, meaning the dynamics mismatch is occurring in the steering model and contact dynamics. In this task, we use a constant estimated dynamics model, but the real physics of the contact include friction effects, deformation, and other unmodeled dynamics that contribute to the model mismatch, resulting in resets.

In this experiment, our MPT planner is running at 5 Hz on an onboard NVIDIA Jetson Orin, running approximately 2100 simulations every 0.2 s. We measure the position of our vehicle and the barrel with motion capture. For our numerical and hardware experiments, we use a value estimate $\hat{V} \equiv 0$. If data is available, an option is to train a neural network value estimator, as in related works in tree search [8].

V. Conclusion

We present MPT, a new receding horizon planning framework that reuses a rich set of information from prior solver iterations to solve challenging planning problems. Our theoretical analysis guarantees the stability of our method and robustness to model mismatch, characterizing the limitations of tree reuse. We use our planner to produce solutions in real-time for a challenging nonprehensile manipulation task to push a target barrel through an obstacle field. We demonstrate the performance improvement of our algorithm against state-of-the-art sampling-based planners, isolating the effect of replanning with partial and complete information reuse. Our results suggest information reuse is an important area of study that can provide significant improvement to a wide variety of algorithms and applications.

ACKNOWLEDGMENT

The authors would like to thank E. S. Lupu, J. A. Preiss, and F. Xie for technical discussions.

References

[1] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved monte-carlo search," *Univ. Tartu, Estonia, Tech. Rep*, vol. 1, pp. 1–22, 2006.

[2] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[4] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 797–822, 2016.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[6] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, pp. 263–282, Springer, 2015.

[7] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3715–3722, 2020.

[8] B. Riviere, W. Hönig, M. Anderson, and S.-J. Chung, "Neural tree expansion for multi-robot planning in non-cooperative environments," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6868–6875, 2021.

[9] J. Ragan, B. Riviere, and S.-J. Chung, "Bayesian active sensing for fault estimation with belief space tree search," in *AIAA Scitech Forum*, p. 0874, 2023.

[10] M. Kobilarov, "Cross-entropy motion planning," *Int. J. Robot. Res.*, vol. 31, no. 7, pp. 855–871, 2012.

[11] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *IEEE Int. Conf. Robot. Automat.*, pp. 1433–1440, 2016.

[12] M. Diehl *et al.*, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *J. Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[13] M. Schneider, "Receding-horizon planning using recursive monte carlo tree search with sparse action sampling for continuous state and action spaces," in *Amer. Control Conf.*, pp. 5401–5406, 2016.

[14] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Trans. Robot.*, 2023.

[15] F. Bertoncelli, F. Ruggiero, and L. Sabattini, "Linear time-varying MPC for nonprehensile object manipulation with a nonholonomic mobile robot," in *IEEE Int. Conf. Robot. Automat.*, pp. 11032–11038, 2020.

[16] G. Zhang, S. Ma, Y. Shen, and Y. Li, "A motion planning approach for nonprehensile manipulation and locomotion tasks of a legged robot," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 855–874, 2020.

[17] M. Selvaggio, A. Garg, F. Ruggiero, G. Oriolo, and B. Siciliano, "Non-prehensile object transportation via model predictive non-sliding manipulation control," *IEEE Trans. Control Syst. Tech.*, 2023.

[18] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang, "Rearrangement with nonprehensile manipulation using deep reinforcement learning," in *IEEE Int. Conf. Robot. Automat.*, pp. 270–277, 2018.

[19] B. V. Mirtich, *Impulse-based dynamic simulation of rigid body systems*. University of California, Berkeley, 1996.

[20] J.-J. E. Slotine and W. Li, *Applied nonlinear control*. Englewood Cliffs, N.J: Prentice Hall, 1991.

[21] T. Lew, A. Sharma, J. Harrison, A. Bylard, and M. Pavone, "Safe active dynamics learning and control: A sequential exploration–exploitation framework," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 2888–2907, 2022.

[22] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *IEEE Int. Conf. Robot. Automat.*, pp. 5883–5890, 2017.

[23] H. Tsukamoto and S.-J. Chung, "Learning-based robust motion planning with guaranteed stability: A contraction theory approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6164–6171, 2021.

[24] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine, "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview," *Annual Rev. Control*, vol. 52, pp. 135–169, 2021.

[25] N. Mao-Lin, "Existence condition on solutions to the algebraic riccati equation," *Acta Autom. Sinica*, vol. 34, no. 1, pp. 85–87, 2008.